A Taste of OFI

Sean Hefty

#OFADevWorkshop
Elegant and intellectually satisfying with subtle undertones?

Selected analysis of the Open Fabrics Interfaces architecture and implementation
Development

Rough conceptual model

Requirement analysis

Input from wide variety of devices

Quarterly release cycle

Deployment

Iterative design and implementation

Collective feedback from OFI IWG

~200 requirements MPI, PGAS, SHMEM, DBMS, sockets, …
Application Requirements

Give us a *high*-level interface!

Give us a *low*-level interface!

And this was just the MPI developers!

Try talking to the government!
Design

- Enable simple, basic usage
- Move functionality under OFI

Implementation Agnostic

- Advanced application constructs
- Expose abstract HW capabilities

Range of usage models
Open Fabrics Interfaces (OFI)

- **Control Services**
  - Discovery
  - `fi_info`

- **Communication Services**
  - Connection Management
  - Address Vectors

- **Completion Services**
  - Event Queues
  - Counters

- **Data Transfer Services**
  - Message Queues
  - Tag Matching
  - RMA
  - Atomics

**OFI Enabled Applications**

- MPI
- SHMEM
- PGAS

**Triggered Operations**
Fabric Information

Endpoint Types
- MSG
  - Reliable connected
- DGRAM
  - Datagram
- RDM
  - Reliable datagram messages
  - Reliable unconnected

Capabilities
- Message queue
  - FIFO
- RMA
- Tagged messages
  - Sends match with specific receive buffers
- Atomics

Select desired endpoint type and capabilities
Fabric Information

OFI Enabled Applications

RDM Message Queue

Common Implementation

DGRAM Message Queue
Fabric Information

- **Capabilities**
  - Application desired *features* and *permissions*
  - Primary capabilities
    - Must be requested by application
  - Secondary capabilities
    - May be requested by application
    - May be offered by provider

- **Attributes**
  - Defines the *limits* and *behavior* of selected interfaces
  - Negotiated

- **Mode**
  - Provider request on application
Threading Options

**Fully thread safe**

**GURU**

Identify resource usage constraints needed for lockless access

Example:

thread 1: \{endpoint 1, CQ 1\}
thread 2: \{endpoint 2, CQs 2-3\}
Progress

What thread context does a request use to complete?

Automatic – submit and forget

Manual – application thread used to complete request

Timeouts and retries, ack processing, atomic operations, data placement, etc.
Resource Management

Enabled – prevent overrunning local and remote queues, including completion queues

Disabled – application responsible for preventing overruns

App: behavior that can be relied on.
Provider: requirements and potential optimizations not restrictions.
Ordering

Sequence of request and completion processing

Strict – requests are processed and completed in order

Relaxed – enable out of order processing
  Dynamic routing, optimized completion processing, parallel data transfers, optimized retry algorithms, etc.
Mode Bits

Provider hints on how it should best be used

None! – provider does all the work!

Application support may improve performance

- Local MR – must register buffers for local operations (send/receives)
- Context – app provides ‘scratch space’ for provider to track request (full or partial onload models)
- Buffer prefix – app provides space for network headers (usnic, IB GRH)
Architecture

Open Fabrics Interfaces (OFI)

Control Services
- Discovery
-fi_info

Communication Services
- Connection Management
- Address Vectors

Completion Services
- Event Queues
- Counters

Data Transfer Services
- Message Queues
- RMA
- Tag Matching
- Atomics

Triggered Operations

OFI Enabled Applications

MPI  SHMEM  PGAS

March 15 – 18, 2015  #OFADevWorkshop
Endpoints

Addressable communication portal

Sequence of request and completion processing

Conceptually similar to a socket or QP
Enable resource manager to direct use of HW resources

Number of endpoints greater than available resources

Map to command queues or HW limits (caching)
Scalable Endpoints

Multiple Tx/Rx contexts per endpoint

- Multi-threading
- Ordering
- Progress
- Completions

Incoming requests may be able to target a specific receive context
OFI Is a Full Meal Deal

• Select a main dish and a side

• OFI tells you today’s specials

• You select the ingredients and tell OFI how to assemble them

We treat you right!

March 15 – 18, 2015
#OFADevWorkshop
OFI 1.0

• Framework – libfabric
  – Interfaces & data structures definitions
  – ‘Spec’ = man pages

• Functional implementation
  – Quickly enable hardware and fabrics
    • Portions layer over vendor interfaces
  – Allow for application development
  – Amount and quality of support is provider specific

Important to distinguish between architecture and direction versus current implementations
OFI 1.0 Providers

- **Sockets**
  - Implement all interfaces and functionality
  - App. development & debug
- **Verbs**
  - Targets any verbs HW
    - Not optimized for a specific device
  - Only common verbs functionality supported
- **PSM**
  - Targets non-verbs HW
  - Expands capabilities beyond lower software driver
- **USNIC**
  - Targets non-verbs HW
  - Cisco will address

Input from verbs derivative and non-verbs providers also fed into OFI design
OFI 1.x

• Address other requirements
  – Multicast
  – Virtualization
  – Features cut from 1.0 release

• Expand and optimize providers
  – Native providers
  – Additional hardware
Thank You